PIG WELFARE THROUGH BEHAVIOR LEARNING FROM CAMERA RECORDINGS

Marin, Iuliana; Goga, Nicolae; Doncescu, Andrei

Published in:
Scientific papers-Series d-Animal science

Publication date:
2018

Scientific Papers. Series D. Animal Science. Vol. LXI, Number 1, 2018
ISSN 2285-5750; ISSN CD-ROM 2285-5769; ISSN Online 2393-2260; ISSN-L 2285-5750

PIG WELFARE THROUGH BEHAVIOR LEARNING FROM CAMERA RECORDINGS

Iuliana MARIN[1], Nicolae GOGA[1,2], Andrei DONCESCU[3]

[1]Faculty of Engineering in Foreign Languages, University POLITEHNICA of Bucharest, Bucharest, Romania
[2]Molecular Dynamics Group, University of Groningen, Groningen, Netherlands
[3]Laboratoire d'analyse et d'architecture des systemes, Université Paul Sabatier de Toulouse, Toulouse, France
Corresponding author email: marin.iulliana25@gmail.com

Abstract

Animal Science students along with the farm staff have to monitor the behavior of pigs in order to assure their welfare. The video systems are used by our educational software and new methods of pig observation, evaluation and treatment are applied much faster and more efficient compared to the classical intervention. Each recording is stored as a media file and each frame taken at 0.1 seconds is stored as a Bitmap image. The Bitmap images are processed in parallel using the MapReduce programming model from Apache Hadoop. The contour of the image is automatically analyzed and based on it the presence of pigs is detected, as well as their location can be determined. The location is important because it can be denoting that the pig eats or that it stays aside. Pig limp was also detected. It was observed based on the recordings that 83% of the time the pigs spend it lying down, 7% is spent eating and 10% of the time they walk and sit. Video monitoring and automatic interpretation facilitates the learning of new intervention approaches and boosts the responsiveness among the students. The students can learn from the critical situations and benefit from these cases while learning.

Key words*: educational software, behavioral monitoring, video recordings, image filtering, MapReduce.*

INTRODUCTION

Image analysis starting from video recordings is a useful way of better understanding the monitored environment.

The use of camera recordings has been used while investigating the way students build their knowledge during the class while the teacher was presenting the lesson (Carvalho, 2004).

In vivo behavioral studies (Pasqualin et al., 2018) have been done in real time based on fast video capturing of dynamic phenomena like fast fluorescent calcium imaging and voltage mapping in cardiac myocytes and neurons.

Pigeons behavior consisting in moving, eating, returning to the box is observed without the need of watching the video (Madan et al., 2014). The algorithm finds out the bird's position at a specific moment of time, overlays this on the reference video frame and a summary of the activity is obtained. Based on this statistics about the length of the path and duration are computed.

Hen tracking based on image analysis processing (Kashiha et al., 2014) is used to determine the total time spent in each compartment, as well as their behavior when they are exposed to a certain ammonia concentration in the air. Ammonia results from the chemical decomposition of uric acid which is found in litter (Aziz et al., 2010). The experiment proved that hens avoid compartments with a level of ammonia higher than 22 ppm.

An automated monitoring system based on using depth sensors (Matthews et al., 2017) for monitoring the pig movement was created for having information about the pig standing, eating, drinking and moving activities.

The depth and infrared captures were done using the Kinect software development kit. The observations on the behavior have been stated manually.

The current article presents the automatic analysis done in parallel of camera recordings to investigate the presence, behavior and limping of pigs. The devices that are needed are just the video cameras. No other special and expensive devices are required. The solution allows the farm staff to assure welfare to pigs.

Students and the farm staff learn to prepare themselves to interfere during an emergency.
The article is organized as follows: in the next section are presented the algorithms for edge detection and the MapReduce parallel processing on images. The results and discussions section presents the impact of computation without and with MapReduce, along with the detection of pig body parts and behavior. The conclusions are drawn based on the obtained results.

MATERIALS AND METHODS

The camera recordings belonging to a pig farm are processed in parallel using the MapReduce programming model from Apache Hadoop. The retrieved media file is the input to be analyzed by the program. Each frame taken at 0.1 seconds is saved in the Bitmap format. The frames are processed in parallel using the C# language based on the MapReduce programming model from Apache Hadoop.
Edges present a significant change in the image intensity, marking the boundaries between the objects. The contours of the pigs can be detected using the Prewitt filter that calculates an estimation of the image intensity function gradient. This offers the direction of the largest possible increase from light to dark and the rate of change in that direction (Prewitt, 1970). The Canny edge detection (Canny, 1986) filter has multiple stages, where edges are the local maxima obtained after applying the horizontal and vertical masks.
The gradient for a slide of the radiography is a two dimensional vector, being composed of the derivatives in the horizontal and vertical directions. The vector points towards the direction of the biggest intensity increase, respectively from darker to brighter values.
There are two masks that are used for edge detection in the horizontal and vertical direction and which are convolved through the operator * with the original image A for which the approximations of the derivatives are done. The vertical mask, Gx, has a zero column and will retrieve the vertical edges of an image:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * A \quad (1)$$

The mask acts as a first order derivative and computed the difference between the intensities of pixels in an edge region. The central column is filled with zeros, indicating that the initial values of the image are not taken into consideration, but it enhances the calculation of the difference between the right and left pixel values placed around an edge.
The horizontal mask, Gy, has a zero row and works similarly to the previous mask, computing the difference of pixel intensities from above and below a particular edge, being equal to:

$$G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * A \quad (2)$$

The marks have opposite signs and their sum equals zero.
The resulting gradient is the combination of the previous two masks:

$$G = \sqrt{G_x^2 + G_y^2} \quad (3)$$

Another type of filter is the Canny edge detector which consists of multiple stages, being an improvement of the Sobel filter (Sobel, 1968). The application of the Gaussian filter to smooth the image in order to remove the noise which is done by blurring the image.
Finding the intensity gradients of the image due to the fact that a point may have a variety of directions. The horizontal and vertical masks are applied, such as for Prewitt which was previously explained.
The application of non-maximum suppression to get rid of response to edge detection. Only local maxima should be marked as edges.
The application of double threshold to determine potential edges. The suppression of all the other edges that are weak and not connected to strong edges. For the second step, the intensity of the gradient can be computed using a Sobel filter, where:

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} * A \quad (4)$$

and

$$G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} * A \quad (5)$$

The gradient may be computed using equation 3 and its direction is determined by the arctangent function with two arguments:

$$\theta = \text{atan2}(G_y, G_x) \quad (6)$$

The MapReduce programming model consists of three steps: Map, Shuffle and Sort, Reduce (Dean, 2008) and was described by the engineers from Google, Jeffrey Dean and Sanjay Ghemawat. The Map step takes as input key-value pairs and outputs multiple pairs. These intermediate output is grouped by key at the Shuffle and Sort step, so that for each key is obtained a list of values and this will represent the input for the Reducer. The Reducer outputs a key-value pair.

MapReduce runs using Java as programming language, but it does not have a library for processing images such as the Bitmap, a raster image file format. Instead, the C# programming language of the .Net Framework contains the System.Drawing.Bitmap library, enhancing image processing. The code written in C# for processing the input data can be run in Java by transforming it into a .dll file which is used for the Map and Reduce functions, but the implementation complexity increases in the generation of the .dll file for the Prewitt and Canny filters.

MapReduce jobs can be run on Hadoop using HDInsight clusters and Microsoft Azure (Microsoft, 2018) that is a cloud computing platform employed to build, deploy and manage applications and services through a network of managed data centers. The Microsoft Azure cloud computing platform requires details about your own credit card and clients pay for the usage according to the pay-as-you-go pricing model.

There exists a MapReduce library called HIPI (Hadoop Image Processing Interface), created by Sean Arietta, Jason Lawrence, Liu Liu, and Chris Sweeney, from the University of Virginia Computer Graphics Lab. It is designed as a wrapper over the Apache's Hadoop MapReduce framework, specifically designed for image processing. It enables the storage of very large collections of images on the HDFS (Hadoop Distributed File System). HIPI integrates OpenCV, a popular open-source library for computer vision algorithms. A HIPI program is structured as in Figure 1.

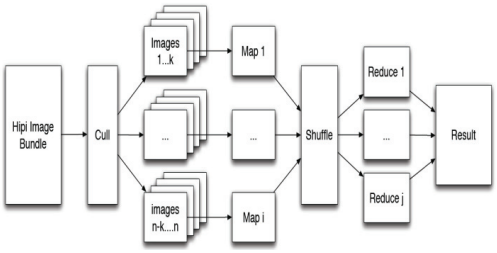

Figure 1. HIPI program design [http://hipi.cs.virginia.edu/images/hipi_pipeline.png, 2018]

A HIPI program takes a HIB (HIP Image Bundle) file as its main input. The custom format encapsulates a collection of images and is a single file on the HDFS.

Fortunately, the standard distribution of HIPI (HIPI, 2018) offers several tools for HIB creation, including a shell script that converts a directory of standard images (png or jpeg) into a HIB file.

The HIPI program's first step is culling, meaning that images from the HIB are filtered by user defined criteria. The remaining images are assigned to maptasks, maximizing data locality (like any Hadoop MapReduce program should). An image is sent to the Mapper as an object (various subclasses exist here, for ease of use). Along with the image, the Mapper is also given a Header object, to uniquely identify each image.

The built-in MapReduce shuffle algorithm minimizes network traffic, while transmitting all output from the Mappers to Reducers. The user can define custom reduce tasks, that provide the building blocks for the final output, stored in the HDFS.

The distribution provides several usage examples (Sweeney, 2018).

As previously mentioned, HIPI can be used with OpenCV. Once you have setup Hadoop and HIPI, the setup for OpenCV is very easy and is described at (Mayank, 2018).

The video recording files are converted into Bitmap images, where each frame is taken as 0.1 seconds. This supports lossless data compression through a program written in C#.

Each slide contains data patches and it is partitioned into four subsets that are further distributed to different map tasks (Fernandez, 2016; Yamamoto, 2012).

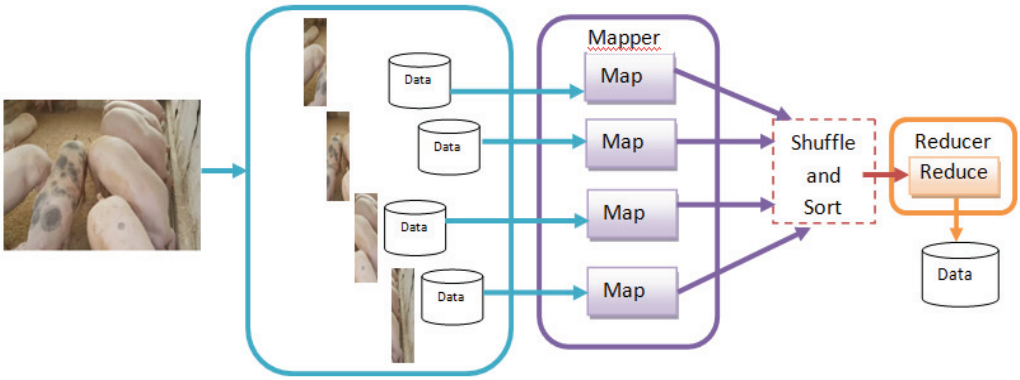

Figure 2. MapReduce processing flow

Parallel processing can be used for computing the performances (Marin, 2010), digestibility (Marin, 2017) of pigs based on forages (Marin, 2016), as well as that of fish (Nicolae, 2017), ducks (Marin, 2015).

The parallel distributed processing using MapReduce is illustrated in Figure 2. The MapReduce function takes as parameters the map function, reduce function and the list of inputs. Each input is transformed into a intermediate result of the type IntermediateResult. The reduce function starts only when all the concurrent transformations have been done. Moreover, it turns the intermediate results into final results. The map task starts by using Task.Factory.StartNew(() => map(input)). The reduce task waits until all the map tasks finished, after which is starts processing using Task.Factory.Continue WhenAll(mapTasks, tasks => PerformReduce (reduce, tasks)).

The input key-value pairs for the Mapper are stored inside a list of MyBitmap images. The list contains pairs of the type MyBitmap where the key is of the type String standing for the image name and the image split representation stored as System.Drawing.Bitmap. The output of the Map function are the intermediate key-value pairs where the key is the image name and the value is the processed subimage.

The Shuffle and Sort step emits key-value pairs where the key is the image name without containing the split number and the value is the list of processed subimages that belong to the same image.

Having this as input, the Reducer will generate MyBitmap key-value pairs where the key is the image name and the value if the final image obtained by merging the filtered subimages.

One important issue that appears is for the border pixels of the subimages, because edge detection algorithms imply convolution operations (Tesfamariam, 2011). According to this, each subimage data patches will be added another layer of pixels from the following subimage as in Figure 3.

Subimage A has an additional border of 2 pixels taken from the following image to its right hand side. The subimages B and C have a border of 2 pixels each taken from their neighbors added on both sides. Subimage D has the additional border of two pixels taken from the left image neighbor. The subimages are overlapping and the Reducer eliminates the added pixels and merges the filtered images according to their name that contains the split number.

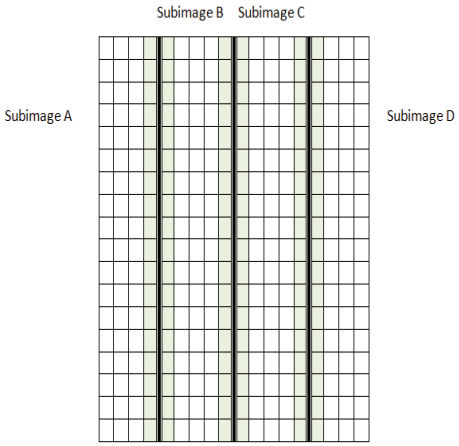

Figure 3. Input image splitting strategy

RESULTS AND DISCUSSIONS

The program runs both without the implemented MapReduce, as well as using it. The input data is represented by 20 folders, each folder containing video recordings. The total size is of 3 GB. The total execution time is illustrated in Table 1.

Table 1. Data execution time according to the input data amount

Number of files	Canny without MapReduce [s]	Prewitt without MapReduce [s]	Canny with MapReduce [s]	Prewitt with MapReduce [s]
840	470.670	315.067	312.810	280.965
630	358.772	237.025	238.874	207.840
420	259.375	152.734	163.102	138.346
210	157.015	68.526	88.293	71.853

Using the data from Table 1, the duration in seconds as function of total file number was displayed in Figure 4.

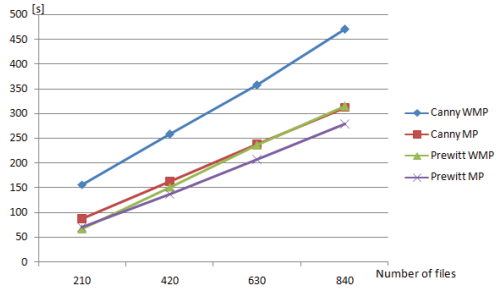

Figure 4. Duration in seconds as function of total file number

It can be observed that the dependency between the duration in seconds and the total number of files is linear.

The execution time without MapReduce data parallel processing lasted longer, as expected, compared to the version where MapReduce was applied.

Taking into consideration the quality of the output image, for the MapReduce filtered scan output Bitmap images of a video recording.

In Figure 5 is illustrated the output triggered by applying the Prewitt filter. For the same slide, the Canny filter was applied and the output looks as in Figure 6.

Figure 5. Application of the Prewitt filter

Figure 6. Application of the Canny filter

By applying the Canny filter, the pigs are even worse to be identified.

Based on the contour which is present in the Prewitt filtered frame, pig recognition is done based on a fragment dictionary for each feature, such as the pig's body, head, legs, including limp.

If limp is detected or if the pigs lay too much down, meaning that they are ill, the image which lead to this diagnosis is sent to the program's user via email to facilitate a faster intervention. In such a situation, the students can learn how to mitigate the problems.

The pigs are analyzed using the recording frame images, where each region is detected by performing a set of transformations, respectively scaling, rotation and translation. A classifier is used in order to trigger which is the detected body part.

Each camera has its own positioning and based on its orientation it is known the location where the pigs eat. The previous detected contours are known and behavioral inferences are triggered based on the positions occupied by the pigs, leading to the fact that 83% of the pigs lay down, 7% eat and 10% change their position frequently.

CONCLUSIONS

The implemented MapReduce accepts Bitmap images as input in order to process them in comparison with the option of using the code written in C# for running it in Java through the obtained .dll file that can be used for the Map and Reduce functions increases the complexity in the generation of the Prewitt and Canny filters.

Another option was that of Microsoft Azure cloud computing platform that asks for your personal information and it uses the pay-as-you-go pricing model.

Large data amounts processing requires the use of parallelization for optimum runtime duration in order to detect illnesses and to analyze the behavior of pigs. This has been proven by the use of the implemented MapReduce, while for the version without it, the execution time was even double.

A better image contour detection and further evaluation is to be investigated in the future.

REFERENCES

Scarfe, W. C., Farman, A. G., 2008. What is cone-beam CT and how does it work? Dent Clin North Am., 52(4), 707-730.

Prewitt, J.M.S., 1970. Object Enhancement and Extraction. Picture processing and Psychopictorics, Academic Press.

Canny, J., 1986. A computational approach to edge detection. Pattern Analysis and Machine Intelligence. IEEE Transactions on PAMI.

Sobel, I., Feldman, G., 1968. A 3x3 Isotropic Gradient Operator for Image Processing. Stanford Artificial Intelligence Project (SAIL).

Dean, J., Ghemawat, S., 2008. MapReduce: Simplified data processing on large clusters. Commun Acm, 51(1), 107-113.

Microsoft, 2018. Microsoft Azure, https://azure.microsoft.com/en-us/?b=16.17 .

HIPI, 2018. HIPI, https://github.com/uvagfx/hipi .

Sweeney, C., Liu, L., Arietta, S., Lawrence, J., 2018. HIPI: A Hadoop Image Processing Interface for Image-based MapReduce Tasks, University of Virginia, http://homes.cs.washington.edu/~csweeney/papers/undergrad_thesis.pdf .

Mayank, S., 2018. How to use opencv with Hadoop or HIPI (Hadoop Image Processing Interface), https://techgimmick.wordpress.com/2015/06/24/how-to-use-opencv-with-hadoop-or-hipi-hadoop-image-processing-interface/ .

Fernandez, B. A., Kumar, S., 2018. Distributed Image Processing using Hadoop MapReduce framework, http://search.iiit.ac.in/cloud/presentations/26.pdf .

Yamamoto, M., Kaneko, K., 2012. Parallel image database processing with MapReduce and performance evaluation in pseudo distributed mode. International Journal of Electronic Commerce Studies, 3(2), 211-228.

Marin, M., Urdes, L., Pogurschi, E., Dragotoiu, D., 2010. Research Concerning the Influence of the Reducing Level of the Compound Feed on the Performances of the Pigs for Fattening. Scientific Papers Animal Science and Biotechnologies, 43 (1), 72-75.

Marin, M., Hodosan, C., Dinita, G., Nicolae, C. G., 2017. The Influence of the Chemical Composition of Maize, Barley and Peas Hybrids on the Digestibility of Compound Feed for Pigs. AgroLife Scientific Journal, 6 (2), 127-132.

Marin, M., Hodosan, C., Nicolae, C. G., Dinita, G., Dragotoiu, T., Nistor, L., 2016. Researches regarding the Chemical Composition and Gross Energy of Sorghum in Comparison to other Forages for Feeding Cattle and Pigs. Scientific Papers, 95.

Nicolae, C. G., Moga, L. M., Bahaciu, G. V., Marin, M. P., 2017. Traceability system structure design for fish and fish products based on supply chain actors needs. Scientific Papers: Series D, Animal Science-The International Session of Scientific Communications of the Faculty of Animal Science, 60.

Marin, M., Dragotoiu, D., Nicolae, C. G., Dinita, G., 2015. Research on the influence of the oregano oil use over the productive performances and quality of duck meat. AgroLife Scientific Journal, 4 (2), 48-51.

Tesfamariam, E. B., 2011. Distributed processing of large remote sensing images using MapReduce: A case of edge detection methods. LAP LAMBERT Academic Publishing.